Trabalho Final – CIC133

Carlos Henrique Reis
Vinicius Manoel Lopes
Lucas Dzin

1. Programação Imperativa

A programação imperativa é o paradigma mais antigo, já que está fundamentado no modelo computacional clássico de “von Neumann Eckert”. Nesse modelo, tanto o programa quanto as suas variáveis são armazenados juntos, e o programa contém uma série de comandos para executar cálculos, atribuir valores a variáveis, obter entradas, produzir saídas ou redirecionar o controle para outro ponto nessa série de comandos. A abstração procedural é um componente para a programação imperativa assim como as atribuições, os laços, as seqüências, os comandos condicionais e a manipulação de exceções. As linguagens de programação imperativa predominantes incluem Cobol, Fortran, C, Ada e Perl.

1.1. Linguagem de Programação C

Para este trabalho será usado a linguagem C, criada em 1972 nos laboratórios Bell por Dennis Ritchie, a linguagem C se tornou uma das mais bem sucedidas linguagens de alto nível já criadas, sendo considerada até hoje, na maioria dos cursos de computação do país, como a linguagem básica para o aprendizado da disciplina. Com um teor de abstração relativamente elevado, ela está mais próxima da linguagem humana do que do código de máquina. Por isso, é considerada por muitos uma linguagem de difícil aprendizado.

1.2. Solução proposta

1.2.1 Backtracking

Os computadores modernos podem utilizar força bruta para resolver problemas de maneira efetiva. Em geral, precisamos desenvolver algoritmos para busca exaustiva em todas as permutações e/ou em todos os subconjuntos possíveis. Os algoritmos desenvolvidos com retrocesso (backtracking) possuem a ideia de criar “cortes” para tornar estes algoritmos mais poderosos.

No problema das oitos rainhas, a ideia é colocar a primeira rainha em uma posição da primeira coluna, a segunda em uma posição da segunda coluna, assim por diante. Cada rainha irá se movimentar na sua coluna até encontrar uma solução. Ao incluir uma rainha deve-se verificar se elas não se atacam, caso elas se atacarem realiza-se o retrocesso (volta para algum estado anterior) para tentar novamente de outra forma.

1.2.2. Algoritmo

A função vallidaPos verifica se ao colocar uma rainha em uma determinada posição, nenhuma rainha atacará, verifica-se a linha, coluna e diagonais a partir da posição onde a rainha será inserida.

```
vallidaPos (linha, coluna)
    se ocorre ataque na linha da nova rainha
         retorna FALSO
     se ocorre ataque na coluna da nova rainha
         retorna FALSO
      se ocorre ataque nas diagonais da nova rainha
         retorna FALSO
      retorna VERDADE
fim
```

A função *executa* gera às soluções, onde *tab* é o tabuleiro, *N* é a dimensão e *col* é a coluna onde vamos tentar inserir a nova rainha

```
executa (tab, n, col)
    se coluna == N
        mostra o tab (solucao)
        retorna
    para i = 0 ate n-1
        se seguro(i, col)
      tab[i][col] = 1//Rainha inserida
            executa(tab, n, col+1)//Para a proxima rainha
            tab[i][col] = 0 //Aqui o retrocesso
fim
```

2. Programação Orientada a Objeto

A programação orientada a objeto (POO) fornece um modelo no qual um programa é uma coleção de objetos que interagem entre si, passando mensagens que transformam seu estado. Neste sentido, a passagem de mensagens permite que objetos de dados se tornem ativos em vez de passivos. Essa característica ajuda a distinguir melhor a programação OO da imperativa. A classificação de objetos, herança e a passagem de mensagens são componentes fundamentais da programação OO. Linguagens orientadas a objetos importantes são Smalltalk, C++, Java e C#.

2.1. Linguagem de Programação JAVA

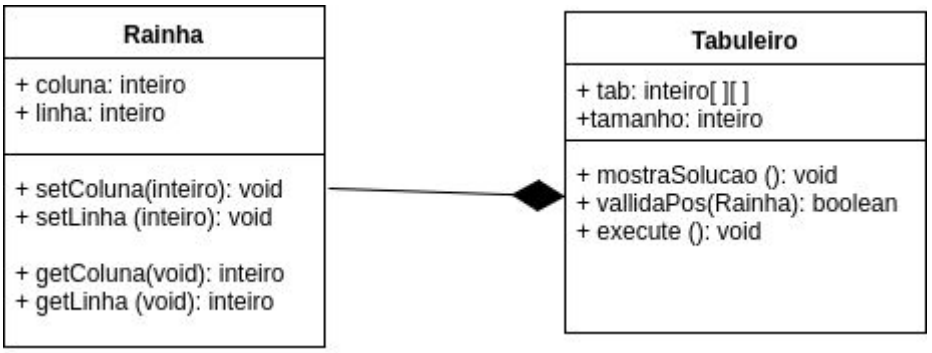
Para a solução do problema usando o paradigma de Orientação objeto foi escolhida a linguagem JAVA, que é uma linguagem de programação orientada a objeto desenvolvida na década de 90. Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um "bytecode" que é executado por uma máquina virtual.

A linguagem Java foi projetada tendo em vista os seguintes objetivos:

- Orientação a objeto
- Portabilidade - Independência de plataforma
- Recursos de Rede
- Segurança

2.2. Solução proposta

No caso foi implementado duas classes: Rainhas e Tabuleiro, onde o Tabuleiro é composto por Rainhas. A seguir veremos o diagrama de classes do problema



A Classe Rainha contém somente dois atributos: coluna e linha, que equivalem a sua posição no tabuleiro. E os métodos get e set dos atributos.

Já a classe Tabuleiro possui o atributo tabuleiro (uma matriz de inteiros), o tamanho do tabuleiro e às rainhas. Além de incluir os métodos mostraSoluçao - Usado toda vez que uma solução é encontrada, validaPos - verifica se ao colocar uma rainha em uma determinada posição no Tabuleiro, nenhuma rainha atacará, e o execute - gera às soluções.

Os métodos da classe Tabuleiro tem comportamentos semelhantes aos descritos na solução proposta na programação Imperativa.

3. Programação Funcional

A programação funcional modela um problema computacional como uma coleção de funções matemáticas, cada uma com um espaço de entrada (domínio) e resultado (faixa). Isso separa a programação funcional das linguagens que possuem o comando de atribuição. As funções interagem e combinam entre si usando composição funcional, condições e recursão. Importantes linguagens de programação funcional são Lisp, Scheme, Haskell e ML.

3.1. Familia de Linguagem LISP

Lisp é uma família de linguagens de programação. O Lisp original foi desenvolvido em 1958 por John McCarthy, então professor no MIT e um dos pioneiros no campo da Inteligência Artificial. Várias idéias utilizadas em linguagens de programação modernas, tais como gerenciamento automático de memória (garbage collection), a estrutura condicional if-then-else e funções recursivas, surgiram com o Lisp.

Com a proliferação da linguagem para diversos tipos de máquinas e sua adoção por diversas instituições, surgiram várias implementações ou dialetos de Lisp. Os dialetos mais usados hoje em dia são o Scheme, criado no período entre 1975 e 1980 por Guy Steele e Gerald Sussman, e o Common Lisp, criado em 1984 com o objetivo de unificar e padronizar as características dos vários dialetos que haviam surgido.

Por simplicidade, costuma-se falar de Lisp como uma linguagem quando se está tratando de características comuns a todos os dialetos de Lisp. Para este trabalho será utilizado o Common Lisp (CLISP).

3.2 Solução proposta

Como nas soluções anteriores usou-se a ideia de Backtracking usando chamadas recursivas da função, e verificando se a rainha pode ou não ficar na posição sem em que se deseja colocá-la.

4. Programação Lógica

A programação lógica (declarativa) permite a um programa modelar um problema declarando qual resultado o programa deve obter, em vez de como ele deve ser obtido. Às vezes, essas linguagens são chamadas de baseadas em regras, já que as declarações do programa se parecem mais com um conjunto de regras ou restrições sobre o problema, em vez de uma seqüência de comandos a serem executados. Interpretar as declarações de um programa lógico cria um conjunto de todas às soluções possíveis para o problema que ele especifi ca. A programação lógica também fornece um veículo natural para se expressar o não-determinismo, o que é apropriado para problemas cujas especificações sejam incompletas. A principal linguagem de programação lógica é o Prolog.

4.1 Linguagem de Programação PROLOG

O PROLOG (PROgramming in LOGic) é uma linguagem interativa criados nos anos 1970, para permitir resolver problemas que envolviam representação simbólica de objetos e seus relacionamentos. Também evita que o programador descreve procedimentos para obter a ParadigmasProgramacao/uploadParadigmasProgramacao/uploadsolução de um problema, permitindo que expresse declarativamente apenas a estrutura lógica do problema através de termos fórmulas atômica e cláusulas.

4.3. Solução proposta

O problema foi modelado considerando que cada rainha era formada por um par com as suas coordenadas (X,Y). A solução final é dada por uma lista de N rainhas que não se atacam. Dado que, para não se atacarem, as rainhas terão de estar colocadas em colunas distintas, podemos fixar as coordenadas X. A solução terá a seguinte forma: [(1,Y1), (2, 2Y), ..., (N, YN)], reduzindo agora o problema é encontrar instâncias deste padrão que não se ataquem.

- Função entre define o intervalo;
- Função member gera os valores;
- Função ataca testa os valores.

5. Comparativo das implementações

Nas implementações em C e Java foram utilizadas estruturas elementares como matrizes e vetores, porém em C foi necessário se preocupar com a parte de alocação de memória, ao contrário do Java que já faz esse tipo de tratamento, além disso em Java foi criado classes

que facilitam no nível de abstração do problema. Em CLisp foi utilizado estrutura do tipo lista e valores numéricos assim como no PROLOG. Em todos os casos fez uso de recursividade para mover as Rainhas no tabuleiro.

Das implementações o CLisp é que possui o menor quantidade de linhas de código, porém, junto com o PROLOG foi o mais difícil de implementar.

Em questão de tempo, para 8 rainhas, não houve uma diferença notável de tempo entre aparas implementações. Já para 9 e 10 rainhas pode-se notar um maior atraso na linguagem Java, se comparado com às demais, esperado por conta da sua máquina virtual